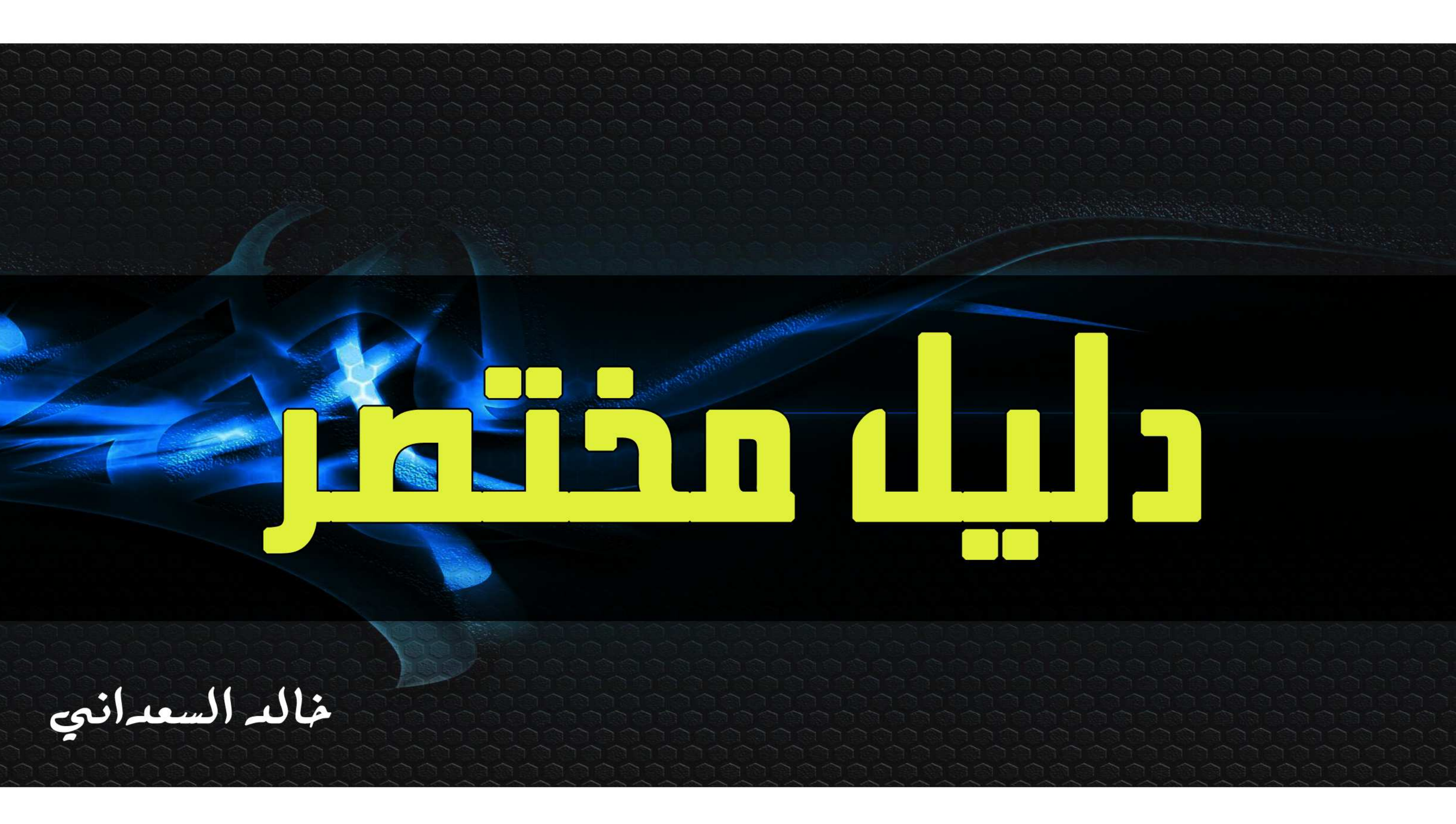
سلسلة كن أسدا للإبداع تقدم:

الهجرة من الفيجوال بسيك إلى الفيجوال سي شارب والعكس.



www.mobarmijoun.com



تقديم

بسم الله الرحمن الرحيم.

عملية الهجرة بين لغات البرمجة أمر شائع عند المبرمجين، وغالبا ما تتخذ هذه الهجرة طابعا تجاريا أو طابعا أكاديميا أو طابعا نفسيا.

المبرمجون الذين ينتقلون من لغة إلى أخرى لأسباب تجارية يبحثون عما يطلبه سوق البرمجيات وكلما وجدوا تدفق الطلبات نحو نوع معين من البرمجيات بحثوا عن التقنيات المستخدمة فيها وعن لغة البرمجة التي تتيح هذه التقنيات فيهاجرون إليها لكي يستخدموها، كمثال على ذلك نجد أن ظهور نظام التشغيل أندرويد الخاص بالهواتف واللوحات الذكية وتزايد الطلبات على التطبيقات والألعاب المبرمجة لهذا النظام، دفعت ثلة من المبرمجين إلى التوجه نحو لغة الجافا أو السي بلس بلس وغيرها...

أما المبرمجون الذين يهاجرون إلى لغة معينة لأسباب أكاديمية ودراسية، فهم يبحثون عن تنمية رصيدهم البرمجي وكلما تعرفوا على لغة برمجية جديدة زادت مداركهم ونمت معارفهم وصاروا أقدر على برمجة ما يطمحون إليه، وغالبا ما يكون هذا الصنف من المبرمجين مبدعا وله مخيلة برمجية واسعة تدفعه إلى ابتكار برامج جديدة لم يسبقه إليها أحد.

النوع الثالث من الهجرة البرمجية الذي يحمل طابعا نفسيا، على ما يلوح لي يكون بسبب عدم الارتياح إلى اللغة البرمجية الأولى، أو السعي إلى تجربة لغات أخرى لسد الهوس البرمجي الذي لم يسلم منه أي مبرمج، هذا الهوس البرمجي هو سبب النظرة الدونية التي يرى بها المبرمج لغته الأولى، فإذا كان يبرمج بإحدى لغات الدوت نيت وسمع بأن لغة الجافا أقوى وأفضل خامره الوسواس فيقرر



الهجرة، وإذا كان يبرمج بلغة الجافا وسمع بأن لغات الدوت نيت تساعد المستخدم على إنتاج برامج بسرعة وأنها تدعم التقارير والطباعة بشكل رائع يقشعر بدنه ويصرخ: يا جافا أنت طالق :)

أحيانا تكون عملية الهجرة بين لغات برمجية تنتمي إلى نفس العائلة لتفوق إحدى هاته اللغات على قريناتها أو بسبب شهرتها وسمعتها.

خلال كتابنا هذا بحول الله سنقوم بعرض أهم الفروق بين لغتي الفيجوال بسيك والسي شارب اللتان تنتميان إلى عائلة الدوت نيت، على أمل أن يجد كل مهاجر دليلا مقتضبا ومختصرا يخول له البرمجة بلغته الجديدة.

هذا الكتاب موجه لمن يريد أن يتحول من لغة الفيجوال بسيك إلى لغة السي شارب والعكس، هو صالح للطرفين معا.



الفهرس


BE LION
arabic formation


BE LION
arabic formation



نبذة عن لغة السي شارب:

كانت فكرة جيدة حينما قررت ميكروسوفت إصدار لغة برمجية جديدة تنتمي لعائلة لغة السي، لأن ذلك سيستهوي فئة عريضة من المبرمجين الذين يستخدمون لغات البرمجة: السي، السي بلس بلس والجافا، وهذا ما كان فعلا، فبمجرد ما أن أطلقت شركة ميكروسوفت أول إصدار من لغة السي شارب جذبت إليها الكثير من المبرمجين وهي حاليا تنافس أقوى لغات البرمجة.

تم إصدار أول نسخة رسمية للغة السي شارب سنة 2002 ضمن إطار العمل Framework 1.0، ومنذ ذلك الوقت واللغة تحقق نجاحا بسبب قوتها المستمدة من عائلة السي، وبسبب سهولتها المستمدة من نطاق الدوت نيت، وبسبب الفئة العريضة التي تستهدفها.

تاريخ لغة الفيجوال بسيك:

الفيجوال بسيك دوت نيت لغة قوية جدا وسهلة الاستخدام، وهي تنتمي لعائلة لغات البسيك Basic وهي اختصار ل Beginner's All-purpose Symbolic Code، وستتفاجأ إن قلت لك بأن تاريخ لغة البسيك التي تنتمي إليها لغتنا يعود إلى سنة 1963.

تتميز لغة الفيجوال بسيك ببعدها عن التعقيدات وميلها نحو التبسيط، من خلالها تستطيع إنشاء برامج ويندوز وبرامج مرتبطة بقواعد بيانات وصفحات الويب، وهي لغة عالية المستوى بعيدة كثيرا في الاصطلاح عن لغة الآلة وقريبة جدا من اللغة الانجليزية.



ظهرت النسخ الأولى من الفيجوال بسيك سنة 1990 مع إصدار Visual Basic 1.0 الذي استمر إلى أن وصل إلى الفيجوال 6، ثم بدأ في الأفول والسقوط، وبمقابل ذلك بدأ الفيجوال بسيك دوت نيت في الشهرة والذيوع بين صفوف المبرمجين بسبب سهولته المستنبطة من عائلة البسيك وبسبب قوته التي يستمدها من نطاق الفريمورك.

أيهما أفضل؟

حسب تجربتي الشخصية فإنني سأتحدث عن اللغتين عمليا بعيدا عن المقارنات البرمجية التي قد لا تجني من ورائها طائلا.

كانت بدايتي مع الفيجوال بسيك وكنت مستمتعا به ولا أزال، بسبب سهولته ووفرة مراجعه ودعمه الشديد لمختلف أنواع المشاريع ابتداء من برامج الشاشة السوداء console، ومرورا بتطبيقات الويندوز وقواعد البيانات والمواقع الديناميكية عبر تقنية asp.net.

ثم بعد ذلك تعرفت على لغة السي شارب وكنت خائفا منها لأنني لدغت سابقا من جحر لغة السي :)

فعلا فمجرد مشاهدة اللامات {} و النقطة الفاصلة ; في شفرات السي شارب يجعلني أتخيل نفسي أنني أمام لغة السي، لكن معرفتي البسيطة بلغتي السي والجافا شجعتني على خوض غمار البرمجة بالسي شارب، فما إن بدأت في البرمجة بها حتى شككت في نفسي: أهذه هي السي شارب التي كنت أخافها؟ لماذا تكذبين يا ميكروسوفت إنها ليست سوى الفيجوال بسيك وقمت بتغيير شكلها :)



بكل صدق، لا أجد فرقا بين لغة السي شارب ولغة الفيجوال بسيك، وأبرمج مشاريعي باللغتين معا بنفس الكفاءة والأداء، وتبقى مسألة الاختيار بين اللغتين مسألة رغبة وميول، فإن كنت أخي الكريم قد درست لغة تنتمي لعائلة لغة السي مثل الجافا والسي بلس بلس، فابدأ بالسي شارب أولا ثم انتقل إلى الفيجوال بسيك إذا أحببت ذلك، وإن كنت قد درست لغة الفيجوال بسيك 6 أو الباسكال أو عندك دراية بلغة VBA المستخدمة في تطبيقات الأوفيس، فإنني أنصحك بالتوجه إلى لغة الفيجوال بسيك دوت نيت ثم انتقل إلى لغة السي شارب إذا أحببت ذلك.

بعد هذه المقدمة، أستطيع الآن أن أبوح لك ببعض الاختلافات بين لغتي الفيجوال والسي شارب:

لغة السي شارب هي لغة كائنية التوجه Oriented Object Programming بشكل كامل، والشفرة دائما تكتب داخل الفئات، بينما تخول لنا لغة الفيجوال بسيك دوت نيت التعامل بكل حرية فهي لغة كائنية التوجه إذا أحببنا التعامل مع الفئات والكائنات وهي أيضا لغة إجرائية Procedural إذا أحببنا التعامل مع Modules والشفرات العادية.

هذا بالإضافة إلى بعض المميزات التي قد توجد في لغة وتغيب في الأخرى، مثل إعادة تعريف الروابط Operator Overloading وهو مدعوم في السي شارب بينما لا وجود له في الفيجوال بسيك، ومثل نافذة إدخال القيم Input Box المدعومة في الفيجوال بسيك والغير موجودة في السي شارب إلا عبر برمجتها من الصفر.

في الختام أحب أن أشير إلى نقطة مهمة وإن شاء الله أكون صادقا فيها: تستطيع برمجة ما تسعى إليه سواء استخدمت الفيجوال بسيك أو الفيجوال سي شارب وهذا يتوقف على مهاراتك وخبرتك البرمجية.



قواعد الكتابة:

في لغة الفيجوال بسيك حينما تقوم بالإعلان عن متغير معين فلست ملزما للانتباه إلى حالة الأحرف لأن ذلك ليس مهما، لكن في لغة السي شارب إحذر فحالة الأحرف مأخوذة بعين الاعتبار.

VB.NET CODE

```
Dim number As Integer
Dim Number As Integer 'Error
```

C#.NET CODE

```
int number;
int Number; //No problem :)
```

هذا بالإضافة إلى بعض القواعد المشتركة بين اللغتين في طريقة الإعلان عن المتغيرات، بحيث لا يقبل أن تبدأ أسماء المتغيرات بأرقام أو برموز مع استثناءات قليلة، أو أن تكون أسماء المتغيرات منتمية إلى الكلمات المحجوزة.

VB.NET CODE

```
Dim 1number as Integer   'Error
Dim $number as Integer   'Error
Dim class as Integer     'Error
```

C#.NET CODE

```
int 1number; //Error
int $number; //Error
int class;   //Error
```



الإعلان عن المتغيرات variables:

نستعمل المتغيرات لكي نخزن بعض البيانات في الذاكرة لنستعملها أثناء تنفيذ البرنامج Runtime، وكل متغير يملك اسما يميزه عن باقي المتغيرات ويملك نوعا Data Type لتحديد طبيعة القيمة المراد تخزينها فيه أهي رقمية أم نصية أم غير ذلك.

VB.NET CODE

```
Dim number As Integer
Dim number2 As Integer = 15
Dim name As String
Dim name2 As String = "Ahmed"
```

C#.NET CODE

```
int number;
int number2 = 15;
string name;
string name2 = "Ahmed";
```



الإعلان عن الثوابت Constants:

الثوابت شبيهة جدا بالمتغيرات من حيث الدور المنوط بها، إلا أنها تختلف عنها في كون قيمة الثابت لا تتغير أبدا عند تنفيذ البرنامج وتبقى ثابتة دائما.

VB.NET CODE

```
Const pi As Double = 3.14
Const state As String = "UnKnown"
```

C#.NET CODE

```
const double pi=3.14;
const string state = "UnKnown";
```

التعليقات Comments:

التعليقات هي عبارات نقوم بكتابتها في برامجنا ويتجاهلها المترجم Compiler لأن دورها يكون فقط من أجل عنونة الكود لتسهيل قراءته أو لتدوين بعض الملاحظات عليه من قبل المبرمج.

VB.NET CODE

```
'This is a single - line comment

'This is
'a multi - line
'Comment
```



C#.NET CODE

```
//This is a single - line comment

/*This is
a multi - line
Comment*/
```

أنواع البيانات Data type:

أنواع البيانات هي طبيعة القيم التي نريد تخزينها في المتغيرات أو الثوابت، توجد العديد من أنواع البيانات منها ماهو نصي string وماهو رقمي integer وماهو منطقي boolean وغير ذلك.

الفيجوال بسيك	الفيجوال سي#
Integer	int
Long	long
Short	short
Byte	byte
Single	float
Double	double
Decimal	decimal
Date	DateTime
String	string
Char	char
Boolean	bool
Object	object



الروابط operators:

الروابط أو المعاملات هي رموز نستخدمها لإجراء بعض العمليات مثل العمليات الحسابية، أو عمليات مقارنة القيم وغير ذلك.

الرابط	الفيجوال بسيك	الفيجوال سي#
روابط العمليات الحسابية		
الجمع	+	+
الطرح	-	-
الجداء	*	*
القسمة	/	/
القسمة الصحيحة الطبيعية	\	/
باقي القسمة	Mod	%
القوة	^	غير موجود لكن توجد دالة لحسابه
روابط إعطاء القيم		
التساوي	`Dim number As Integer` `number = 10`	`int number;` `number = 10;`
الزيادة	`Dim number As Integer` `number += 10`	`int number;` `number += 10;`
النقصان	`Dim number As Integer` `number -= 10`	`int number;` `number -= 10;`
المضاعفة	`Dim number As Integer` `number *= 10`	`int number;` `number *= 10;`



الاختزال	Dim number As Integer number /= 10	int number; number /= 10;
الاختزال الصحيح الطبيعي	Dim number As Integer number \= 10	int number; number /= 10;
دمج النصوص	Dim name As String="Ahmed" name &=" is a teacher"	string name="Ahmed"; name +=" is a teacher";
روابط المقارنة		
أكبر من	>	>
أكبر من أو يساوي	>=	>=
أصغر من	<	<
أصغر من أو يساوي	<=	<=
يساوي	=	==
لا يساوي	<>	!=
الروابط الشرطية		
رابط المعية (و)	and	&&
رابط الاختيار (أو)	Or	||
الزيادة والنقصان بواحد		
الزيادة بواحد	غير موجود	Number++
النقصان بواحد	غير موجود	Number--



البنية الشرطية Flow control:

نستخدم البنيات الشرطية من أجل التحقق من نتيجة شرط معين، وبناء على هذا التحقق ننفذ شفرة معينة دون أخرى.

باستخدام if...else

VB.NET CODE

```
If job = "Doctor" Then
    state = True
ElseIf job = "Nurse" Then
    state = True
Else
    state = False
End If
```

C#.NET CODE

```
    if (job == "Doctor")
    {
        state = true;
    }
    else if (job == "Nurse")
    {
        state = true;
    }
    else
    {
        state = false;
    }
}
```



باستخدام فصل الحالات:

VB.NET CODE

```
Select Case op
    Case "+"
        result = 5 + 5
    Case "-"
        result = 5 - 5
    Case "*"
        result = 5 * 5
    Case "/"
        result = 5 / 5
    Case Else
        result = 0
End Select
```

C#.NET CODE

```
switch (op)
{
    case '+':
        result = 5 + 5;
        break;
    case '-':
        result = 5 - 5;
        break;
    case '*':
        result = 5 * 5;
        break;
    case '/':
        result = 5 / 5;
        break;
    default:
        result = 0;
        break;
}
```



باستخدام المعامل الشرطي:

المعامل الشرطي يستخدم للقيام بعملية من بين عمليتين بعد تحقق شرط معين.

VB.NET CODE

```
Dim access_state As String = IIf(password = "pass123", "correct",
"incorrect")
```

C#.NET CODE

```
string access_state=password=="pass123" ? " correct": "incorrect";
```

البنية التكرارية Loops:

نستخدم البنية التكرارية لتكرار جزء معين من الشفرة عدة مرات.

باستخدام for :

VB.NET CODE

```
For number As Integer = 0 To 9
    Console.WriteLine("Number : "& number)
Next
```

C#.NET CODE




```
for (int number = 0; number < 10; number++)
{
    Console.WriteLine("Number : "+ number);
}
```

باستخدام do :

VB.NET CODE

```
Dim number As Integer = 0
Do Until number > 5
    Console.WriteLine("Line:{0}", number)
    number += 1
Loop
```

C#.NET CODE

```
int number = 0;
do
{
    Console.WriteLine("Line:{0}", number);
    number++;
}
while (number <= 5);
```



باستخدام while :

VB.NET CODE

```
Dim number As Integer = 0
While number < 9
    Console.WriteLine("Line :{0}", number)
    number += 1
End While
```

C#.NET CODE

```
int number = 0;
while (number < 9)
{
    Console.WriteLine("Line :{0}", number);
    number++;
}
```

باستخدام for each :

VB.NET CODE

```
Dim name As String = "Ahmed"
For Each letter As Char In name
    Console.WriteLine(letter)
Next
```

C#.NET CODE




```
string name = "Ahmed";
foreach (char letter in name)
{
    Console.WriteLine(letter);
}
```

باستخدام goto:

VB.NET CODE

```
        Dim founder As String
ask:    Console.Write("who is the founder of microsoft?")
        founder = Console.ReadLine()
        If founder = "bill gates" Then
            Console.Write("You're right")
        Else
            Console.WriteLine("False !!")
            GoTo ask
        End If
```

C#.NET CODE

```
        string founder;
    ask: Console.Write("who is the founder of microsoft?");
        founder = Console.ReadLine();
        if (founder == "bill gates")
        {
            Console.Write("You're right");
        }
        else
        {
            Console.WriteLine("False !!");
            goto ask;
        }
```



الوظائف Methods:

الوظيفة هي مجموعة من الأوامر المجمعة تحت اسم معين، وعند النداء عليها بهذا الاسم يتم تنفيذها.

إنشاء الوظائف:

VB.NET CODE

```
'Create Procedure
Public Sub myProc()
    'Do Something
End Sub

'Create Function
Public Function myFunction() As String
    Return "Something"
End Function
```

C#.NET CODE

```
//Create Procedure
public void myProc()
{
    //Do Something
}

//Create Function
public string myFunction()
{
    return "Something";
}
```



استدعاء الوظائف:

VB.NET CODE

```
myProc()

Dim value As String = myFunction()
```

C#.NET CODE

```
myProc();

string value = myFunction();
```

تمرير البرامترات:

VB.NET CODE

```
'passing parameters by value
Public Function myFunction(ByVal a As Integer, ByVal b As
Integer) As Integer

    Return a + b

End Function

    'Calling the method
    Dim sum As Integer = myFunction(10, 35)
```

C#.NET CODE

```
//passing parameters by value
public static int myFunction(int a, int b)
{
    return a+b;
```




```
        }

        //Calling the method
        int sum = myFunction(10,35);
```

VB.NET CODE

```
    'passing parameters by reference
    Public Function myFunction(ByRef a As Integer, ByRef b As
Integer) As Integer

        Return a + b

    End Function

        'Calling the method
        Dim x As Integer = 3, y As Integer = 9
        Dim sum As Integer = myFunction(x, y)
```

C#.NET CODE

```
        //Passing by reference
        public static int myFunction(ref int a, ref int b)
        {
            return a+b;
        }

        //Calling the method
        int x = 3, y = 9;
        int sum = myFunction(ref x,ref y);
```



المصفوفات arrays:

المصفوفة هي مجموعة من المتغيرات التي تحتوي على نفس نوع البيانات.

VB.NET CODE

```
'Declaring and Poulating an array
Dim colors() As String = New String() {"red", "green", "blue"}
Dim numbers() As Integer = {1, 2, 3, 4, 5, 6, 7, 8, 9}
```

C#.NET CODE

```
//Declaring and Poulating an array
string[] colors = new string[] { "red", "green", "blue" };
int[] numbers = { 1, 2, 3 , 4 , 5 , 6 , 7 , 8 , 9 };
```

التراكيب structures:

نستخدم التراكيب من أجل إنشاء أنواع مركبة قابلة لاحتواء أكثر من نوع بيانات، مثلا تستطيع حفظ معلومات طالب معين (الاسم، العمر، العنوان، ...) في نفس التركيب بدل الإعلان عن متغيرات متفرقة، ويمكن للتركيب أن يضم كذلك وظائف وخصائص بالإضافة إلى الحقول.



VB.NET CODE

```
'Creating the structure
Structure student
    Public name As String
    Public age As Integer
    Public address As String

    Public Function ShowStudent() As String
        Return "Student name:" & name _
               & "Student age: " & age _
               & "Student address: " & address
    End Function
End Structure

    'Creating a new instance of the student structure
    Dim student1 As New student
    'initialize fields
    student1.name = "Khalid"
    student1.age = "24"
    student1.address = "Morocco"
    'access to its methods
    student1.ShowStudent()
```

C#.NET CODE

```
//creating the student structure
struct student
{
    //Fields
    public string name;
    public int age;
    public string address;

    //Methods
```




```
        public string ShowStudent()
        {
            return "Student name: " + name
                 + " Student age: " + age
                 + " Student address: " + address;
        }
    }

        //Creating a new instance of the student structure
        student student1 = new student();
        //initialize fields
        student1.name = "Khalid";
        student1.age = 24;
        student1.address = "Morocco";
        //access to its methods
        student1.ShowStudent();
```

المعددات Enumerations:

نستخدم المعددات إذا كنا نريد تحديد قيم محددة ليتم تخزينها في متغير ما، مثلا حينما نريد حفظ أيام الأسبوع في برنامجنا، فنحن نعلم مسبقا أن مجال أيام الأسبوع محدد والقيم معروفة لذلك يمكننا استخدام المعددات.

VB.NET CODE

```
'Creating the enumeration
Enum WeekDays
    Monday
    Tuesday
    Wednesday
    Thursday
```

```
        Friday
        Saturday
        Sunday
    End Enum
```

C#.NET CODE

```
    //Creating the enumeration
    enum WeekDays
    {
        Monday,
        Tuesday,
        Wednesday,
        Thursday,
        Friday,
        Saturday,
        Sunday
    }
```

معالجة الاستثناءات Handling Exceptions:

الاستثناء ات هي أخطاء تحدث عند اشتغال البرنامج أي في مرحلة التنفيذ Runtime، ويكون ذلك لأسباب عديدة، من بينها إدخال قيمة غير مناسبة، كمحاولة تخزين قيمة نصية في متغير رقمي، أو محاولة القيام بعملية القسمة على صفر، أو محاولة حذف ملف غير موجود أساسا، وغير ذلك...

حينما يحدث استثناء في البرنامج فإنه يتوقف فورا عن الاشتغال، لذلك وجب إدارة هذه الأخطاء عبر استخدام try...catch.

VB.NET CODE

```
        Dim age As Integer
```




```
Console.WriteLine("Enter your age:")
Try
    age = Console.ReadLine
Catch ex As Exception
    Console.WriteLine(Err.Description)
Finally
    Console.WriteLine("press any key to leave...")
End Try
Console.ReadKey()
```

C#.NET CODE

```
int age;
Console.Write("Enter your age:");
try
{
    age = int.Parse(Console.ReadLine());
}
catch (Exception Err)
{
    Console.WriteLine(Err.Message);
}
finally
{
    Console.WriteLine("press any key to leave...");
}
```



الفئات والكائنات Classes and Objects:

الفئات هي أنواع نقوم بإنشائها من أجل تمثيل شامل لكائن معين، وهي شبيهة بالتراكيب structure، أما الكائنات فهي نسخ نقوم بإنشائها من فئة معينة لكي نستعملها في برنامجنا، للإعلان عن الفئات في لغتي الفيجوال بسيك والسي شارب فإننا نستعمل الكلمة المحجوزة class.

VB.NET CODE

```
'Creating a class
Public Class Car
    'Class members
End Class

'Creating new instance named 'object1' of Car class
Dim object1 As New Car
```

C#.NET CODE

```
    //Creating a class
    public class Car
    {
        //Class members
    }

    //Creating new instance named 'object1' of Car class
    Car object1 = new Car();
```



Me و this:

لكي نتمكن من الدخول إلى عناصر الفئة الحالية، نستخدم الكلمة Me في الفيجوال بسيك و الكلمة this في السي شارب، ويمكننا الاستغناء عنهما وكتابة اسم العنصر مباشرة (المقصود بالعنصر حقول ووظائف وخصائص الفئة.

VB.NET CODE

```
Public Class Employee
    Private name As String

    Public Function Work() As String
        Return "I am a developer"
    End Function

    Public Sub initialize(Name As String)
        Me.name = Name
        Me.Work()
    End Sub

End Class
```

C#.NET CODE

```
public class Employee
{
    private string name;

    public string Work()
    {
        return "I am a developer";
    }

    public void initialize(string Name)
```




```
    {
        this.name = Name;
        this.Work();
    }

}
```

الوراثة Inheritance:

الوراثة هي عملية نقل عناصر فئة معينة تسمى الفئة الرئيسية Main class إلى فئة أخرى أو أكثر وتسمى الفئة البنت أو الفئة المشتقة Derived class.

مثلا أستطيع أن أنشىء فئة أسميها Animal، ثم أشتق منها فئات أخرى (أسد، نمر، ...)، وبمجرد القيام بعملية الوراثة فإن عناصر الفئة الأم (الفئة Animal) تنتقل إلى الفئات البنات.

VB.NET CODE

```
'Main class
Public Class Animal
    Private name As String

    Public Function getName() As String
        Return name
    End Function
End Class

'Derived Classes
Public Class Lion
    Inherits Animal
End Class
```




```
Public Class Tiger
    Inherits Animal
End Class

Public Class Horse
    Inherits Animal
End Class
```

C#.NET CODE

```
//Main class
public class Animal
{
    private string name;

    public string getName()
    {
        return name;
    }
}

//Derived Classes
public class Lion:Animal
{
}

public class Tiger : Animal
{
}

public class Horse : Animal
{
}
```



الواجهات Interfaces:

في لغتي الفيجوال بسيك والسي شارب لا يوجد مفهوم الوراثة المتعددة التي تمكننا من وراثة أكثر من فئة مرة واحدة، وتم استبدال هذا المفهوم بالواجهات interfaces، وهي عبارة عن هيكل تقوم الفئات باستعماله بحيث لا تحتوي الواجهات على شفرة معينة وإنما تحتوي فقط على هيكل يضم تعريف لوظائف وخصائص وحقول من دون كود، وحينما تتم عملية استعمال هذه الواجهة من طرف فئة معينة نقوم بإعطاء محتوى لعناصرها. تسمى عملية استعمال الواجهات من طرف الفئات ب implementation.

لإنشاء الواجهات نستخدم الكلمة المحجوزة interface:

VB.NET CODE

```
'Defining ILocation interface
Public Interface ILocation
    Sub setLocation(ByVal x As Integer, ByVal y As Integer)
    Function GetLocation_X() As Integer
    Function GetLocation_Y() As Integer
End Interface

'Implementing ILocation interface
Public Class myForm
    Implements ILocation
    Private x, y As Integer

    Sub setLocation(ByVal X As Integer, ByVal Y As Integer)
Implements ILocation.setLocation
        Me.x = X
        Me.y = Y
```




```
    End Sub

    Function GetLocation_X() As Integer Implements
ILocation.GetLocation_X
        Return x
    End Function

    Function GetLocation_Y() As Integer Implements
ILocation.GetLocation_Y
        Return y
    End Function
End Class
```

C#.NET CODE

```
//Defining ILocation interface
public interface ILocation
{
    void setLocation(int x, int y);
    int GetLocation_X();
    int GetLocation_Y();
}

//Implementing ILocation interface
public class myForm:ILocation
{
    private int x, y;
    void ILocation.setLocation(int X, int Y)
    {
        this.x = X;
        this.y = Y;
    }

    int ILocation.GetLocation_X()
    {
        return x;
    }
```




```
    int ILocation.GetLocation_Y()
    {
        return y;
    }
}
```

المشيدات Constructors:

المشيد هو عبارة عن وظيفة خاصة لا تعيد لنا شيئا وليس لها نوع، ودروه يتجلى في إعداد الكائنات التي سيتم استنساخها من الفئة كإعطاء قيم بدئية لحقول الكائن.

VB.NET CODE

```
Public Class Employee
    Private name As String
    Private age As Integer

    'Constructor Without Parameters
    Sub New()
        Me.New("name not found", 0)
    End Sub

    'Constructor With Parameters
    Sub New(Name As String, Age As Integer)
        Me.name = Name
        Me.age = Age
    End Sub

End Class
```



C#.NET CODE

```
public class Employee
{
    private string name;
    private int age;

    //Constructor Without Parameters
    public Employee()
        : this("name not found", 0) { }


    //Constructor With Parameters
    public Employee(string Name, int Age)
    {
        this.name = Name;
        this.age = Age;
    }

}
```

الخصائص Properties:

الخصائص عبارة عن وظائف نستخدمها للوصول إلى حقول الفئة الحالية من خلال فئات أخرى وكأن هذه الحقول معرفة ب public.

وهي تتكون من جزءين، جزء يسمح بقراءة قيمة الحقل ويسمى Getter، وجزء يسمح بتعديل قمية الحقل ويسمى Setter.

VB.NET CODE




```
Public Class Employee
    Private name As String
    Private age As Integer

    Public Property name_property()
        Get
            Return Me.name
        End Get
        Set(value)
            Me.name = value
        End Set
    End Property

    Public Property age_property()
        Get
            Return Me.age
        End Get
        Set(value)
            Me.age = value
        End Set
    End Property

End Class
```

C#.NET CODE

```
public class Employee
{
    private string name;
    private int age;


    public string name_property
    {
        get { return name; }
        set { name = value; }
    }
```




```
    public int age_property
    {
        get { return age; }
        set { age = value; }
    }
}
```

إنهاء الكائنات:

نحتاج أحيانا إلى إنهاء الكائنات ووضع قيمة "لا شيء" فيها.

VB.NET CODE

```
Dim emp As New Employee
emp = Nothing
```

C#.NET CODE

```
Employee emp=new Employee();
emp=null;
```

مجالات الأسماء namespaces:

لكي نقوم بجلب مجالات الأسماء فإننا نستخدم الكلمة Imports في لغة الفيجوال بسيك، بينما نستخدم الكلمة using في لغة السي شارب.

VB.NET CODE

```
Imports System
Imports System.Collections.Generic
Imports System.Linq
```




```
Imports System.Text
Imports System.Threading.Tasks
```

C#.NET CODE

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

إظهار علبة الرسائل:

لإظهار علبة الرسالة بالشكل التالي:



VB.NET CODE

```
MsgBox("السلام عليكم", MsgBoxStyle.Information, "رسالة")
```

C#.NET CODE

```
MessageBox.Show("السلام عليكم", "رسالة", MessageBoxButtons.OK,
                                  MessageBoxIcon.Information);
```



العناصر الثابتة static members :

العناصر الثابتة هي عناصر لا يمكن الوصول إليها من خلال الكائنات المستنسخة من فئة معينة، وإنما نستطيع الوصول إليها مباشرة بعد كتابة اسم الفئة.

VB.NET CODE

```
Public Class Employee
    'static field
    Public Shared name As String

    'static method
    Public Shared Function getName()
        Return name
    End Function

End Class

Sub Main()
    'use static members
    Employee.name = "Khalid"
    Dim myName As String = Employee.getName
End Sub
```

C#.NET CODE

```
public class Employee
{
    //static field
    public static string name;
```




```
    //static method
    public static string getName()
    {
       return name;
    }

}

      static void Main(string[] args)
      {
            //use static members
             Employee.name = "Khalid";
             string myName = Employee.getName();
      }
```



خاتمة :

انتهينا بحمد الله من عرض أهم المفاهيم البرمجية بلغتي الفيجوال بسيك والسي شارب، ومع ذلك يبقى مجهودنا مجهودا محدودا يشوبه النقص بسبب عجز منا أو سهو وغفلة، لذلك سأمدكم ببريدي الالكتروني وبصفحتي على الفيسبوك وبموقع أكاديمية المبرمجين العرب، لكي ترشدوني إلى ما سهوت عنه لأضيفه في إصدار جديد، وأيضا لكي أستقبل أسئلتكم وأرد عليها بما علمني الله عز وجل.

أدعو لكم بالتوفيق والسداد ودام لكم البشر والفرح والسلام عليكم ورحمة الله وبركاته.

أكاديمية المبرمجين العرب:

http://www.mobarmijoun.com

البريد الالكتروني:

Khalid_ESSAADANI@Hotmail.fr

ESSAADANIKHALID@Gmail.com

الفيسبوك:

www.facebook.com/ESSAADANI.Khalid

www.facebook.com/Khotwa.Amam